

**Nuclear Physics Division**

***Fast Electronics Group***

# Firmware for

**VXS Crate Trigger Processor Version 2 (CTPV2) Spectrometer Module**

###### May 15, 2014

# Hai Dong

# Table of Contents

# Section Title Page

**1.0 Introduction 3**

**2.0 FX70T (U1, U3) VHDL Code**

**3.0 FX100T (U24) VHDL Code**

**4.0 I-SQUARE-C Code**

**5.0 Total Delay**

**6.0 LED**

**7.0 SD Test Code**

**8.0 Configuring FPGA from I-SQUARE-C Bus**

**9.0 Board Serial Number**

**9.0 Register File**

**1.0 Introduction**

The firmware for the Crate Trigger Module (CTP) is written in VHDL. The CTP module has two XC5VLV50 (U1, U3) that has same VHDL code and one XC5VLV110 (U24) that has its own VHDL code. LV50 VHDL code receives data from five FADC, aligns the data, and sends the data in sync with system clock to the LV110. The LV110 receives data from six FADC and both LV50, aligns the data, processes the data, and sends the result to Fiber located at the front panel.

This firmware has the addition of BCAL processing described in Section 12.0.

1. **FX70T (U1,U3) VHDL Code**
   1. **Receive data from FADC**

The VHDL code receives data from five FADC using Xilinx Aurora operating at 2.5Gbits, aligns the data, and sends the data 80 bits synchronized with 250MHz clock. The Aurora protocol provides control and interface signals: Lane\_up, Channel\_up, Rx\_src\_rdy\_n, Tx\_src\_rdy\_n, Tx\_dst\_rdy\_n. The functions for these signals are as follow:

* Lane\_up : indicates FADC MGT and CTP GTP are able to send and receiving data from each other.
* Channel\_up: indicates the data from the lanes if more than one are in alignment.
* Tx\_dst\_rdy\_n: indicates that the transmitter is ready to send data
* Tx\_src\_rdy\_n: tell the transmitter to send user data.
* Rx\_src\_rdy\_n: indicates that the receiver has user data
  1. **Aligning Data**

In the 12 Gev Trigger System, Sync is distributed in time to all sixteen FADC250V2 and CTP modules in a crate. When Sync goes high, FADC250 stops sending data and CTP reset all its circuitries and buffers. Sync has to be high for at least 500 nS to allow the MGT to completely flush its buffers. When Sync goes low, the FADC250 send value 2 for one 16 bits and value 1 for the other 16 bits on the 2 MGT lanes 0 and 1. Values 2 and 1 are sent 3 times. Which value is sent on lane 0 and 1 is depend on when the Sync arrived at the FADC250. The CTP uses these values to align data from all 16 FADC250 in time and words order. After aligning the data from 5 FADC250, U1 sends data to U24 via 80 LVDS lines. U3 does the same as U1. U24 aligns the data from 6 FADC250, wait for data from U1 and U3, processes, and sends data to SSP through fiber. First Word Fall Through FIFO is used to used to indicate when first values 2 and 1 are received.This process is enable when bit 3 of Config1 is a one.

**Demultiplexed Data** **to VLX110 (U24)**

The 32-bits data from each FADC is de-multiplexed into two 16-bits words with bits 31-16 send to U24 first. The 16-bits word is registered (F1\_F3\_D ) and sends to U24 along with DatRdy. After Sync goes low, DatRdy goes high along with all values 2 output. DatRdy goes low when Sync goes high. The 250 MHz input clock is routed inside the FPGA to output pins (CLK\_FPGA1\_3). The LV110 uses CLK\_FPGA1\_3 to capture F1\_F3\_D and Datrdy.

* 1. **Disable FADC**

Data from FADC can be disabled by writing to Configuration register via I-square-C interface. When disabled, the code does not wait for data from that FADC250. The bits corresponded to that FADC are zero going.

* 1. **Processing Delay**
  2. **Xilinx System Monitor.**

Xilinx System Monitor feature that indicates the die temperature and VCCINT and VCCAUX Supply is included. The data is mapped into register file that can be read back through I-Square-C. The host has to convert binary data to temperature and voltage (See Xilinx UG192).

1. **FX100T (U24) VHDL code**
   1. **Receive data from FADC**

The procedure for receiving data from six FADC is identical to U1 and U3.

* 1. **Receive data from U1 and U3**

When Datrdy is high, 80 bits data is clocked into 512\_80 FIFO. Five-hundred words deep can handle 2uS skewing between the two VLX50. The number of clock from Datrdy going high to data FIFO empty going low is 8 (32 ns).

* 1. **Aligning Data.**

Aligning data from six FADC250 is the same as U1 and U3. When Datrdy from U1 goes high, U24 allows words from U1 to be written into First-Word-Fall-Through FIFO. It does the same with U3. When words values of 2 are at the output of these FIFO, the FIFO are read at the same time. This process is enable when bit 3 of Config1 is a one.

* 1. **Summing Data.**

The data is de-multiplexed and add in stages as shown below. The stages are used to make 250 MHz clocking using Xilinx FPGA lowest cost (speed grade 1).

* + 1. First stage
       - 1st\_A0 = VLX50 #1 bit (15..0), (31..16), (47..32)
       - 1st\_A1 = VLX50 #1 bit (63..48), (79..64)
       - 2nd\_A0 = VLX50 #2 bit (15..0), (31..16), (47..32)
       - 2nd\_A1 = VLX50 #2 bit (63..48), (79..64)
    2. Second stage
       - 1st\_B = 1st\_A0 + 1st\_A1
       - 2nd\_B = 2nd\_A0 + 2nd\_A1
       - SUM\_A = FADC\_0 + FADC\_1 + FADC\_2
       - SUM\_B = FADC\_3 + FADC\_4 + FADC\_5
    3. Third stage
       - 1st\_2nd = 1st\_B + 2nd\_B
       - SUMA\_B = SUMA + SUMB
    4. Fourth stage
       - Final\_SUM = 1st\_2nd  + SUMA\_B
  1. **Multiplexing Final SUM**

In the CTP the FIBER data consisted of 4 Aurora Lanes operating at 2.5Gibits. Each lane provides 16bits for a total of 64-bits every 8 NS. Since the 20-bits Final Sum is occurring at 4 NS, two Final Sum is packed (multiplexed) into one transmission. Final Sum # even is packed into bits 39 to 30 while Final Sum # odd is packed into bits 19 to 0. As of this date 03/10/09 bits 63..40 are set to zeroes.

* 1. **Disable FADC**

An FADC can be Data from FADC can be disabled by writing to Configuration register via I-square-C interface. When disabled, the code does not wait for data from that FADC. The bits corresponded to that FADC are zero in the summing process.

* 1. **Processing Delay**

The number of clock from Datrdy (from VLX50) going high to FIFO empty going low is 8 clock (32 nS). Adder stages take 11 clock (44 ns).

* 1. **Threshold**

A register accessed through I-Square-C provides a threshold for the final sum . When the final sum is above the threshold, Trigger Output at the front panel is active high. If the History Buffer described below is armed, 256 additional sums are stored and ready to be read out.

* 1. **History Buffer**

The History Buffer stores 512 Spectrometer Counter HitBits pattern that go to the SSP. The History continues to store 255 samples. When it is trigger, it will store 255 more samples and then stop. It is trigger when a bit that is enable by U24 registers 8 and 9 is a one. A HitBit is enable to trigger the History Buffer when the corresponding bit in U24 Register 8 and 9 is a one. It is controlled and read out via I-Square-C. The steps to use the history buffer are as follow:

* Host sets ARM to one.
* Host sets ARM to zero.
* After 256 Final Sum are stored in Memory, waiting for HitBit Pattern to match U24 Regiser 8 and 9. Continues storing Final Sum
* When Final Sum reach threshold. 256 more values are stored.
* Stops storing.
* Host polled for DataReady for high.
* Read as much values as host wanted. Data read back is auto increment.
* Host sets ARM to clear DataReady.
  1. **Automate Testing.**

The integrity of the data transfer from FADC250 to CTP can be verified. When bit 1 of MGT\_CTRL register in FADC250 is 0, counting sequences (0,1,2,3,4, etc,). The CTP adds the data from all enable FADC250 and compare to expected sum. For example, if 8 FADC250 are enable, the expected Sum are 0, 8, 16, 24, 32, etc. If at any time the actual Sum and the expected Sum are not equaled, ERROR\_LATCH\_FS variable goes high and stay high until Sync goes high. To use this feature:

* Reset control\_mgt bit 3 so the FLASH is sending SUM to CTP
* **Execute all steps except the last step (10.11) in “Steps to Align Data from FLASH”. Leave FADC250** control\_mgt bit 1 a zero..
* Wait for however long the testing is desire.
* Read Is2 status register bit ERROR\_LATCH\_FS. If it is zero the data from the FADC250 are OK.
* Make bit 1 of of MGT\_CTRL register in FADC250 one. ERROR\_LATCH\_FS should be one because data from FADC250 is not counting sequence.
  1. **Xilinx System Monitor**

Xilinx System Monitor feature that indicates the die temperature and VCCINT and VCCAUX Supply is included. The data is mapped into register file that can be read back through I-Square-C. The host has to convert binary data to temperature and voltage (See Xilinx UG192).

1. **I-Square-C interface**

Data is written to and read from the CTP serially from VXS P1 connector. The serial interface consists of an input clock bit and a bidirectional data bit that connected to all three FPGA. The FPGA de-serializes the data bits and listens for its address. The FPGA have non-overlapping address and responses when it is addressed. Each Register is 16 bits wide.

1. **Total Processing Delay**

It takes a total of 27 clocks (8 from VLX50, 8 to receive data from VLX50, 11 from adder stages) from the time data arrived from the slowest FADC to produce the first Final Sum.

1. **LED**

LED(0) <= HEART\_BEAT\_250 ;

LED(1) <= HEART\_BEAT\_200;

LED(2) <= ABOVE\_THREDSHOLD\_BUF\_Q(0); -- ; Heartbeat 200

LED(3) <= indicate all MGT lanes and all MGT Channels of U24 are OK.

LED(4) <= HEART\_BEAT\_INIT;

LED(5) <= Indicate there is not Fiber Fault and Fiber is ready---

LED(6) <= ‘1’;

1. **SD Test Code**
   1. Count Rising Edge of Sync, Trig1, Trig2 from SD.
   2. Measure Clock from SD (should be 250 MHz)
   3. Connect SD\_IN\_SPARE 0 to SD\_OUT\_SPARE 0
   4. Connect SD\_IN\_SPARE 1 to SD\_OUT\_SPARE 1
2. **Configuring FPGA from I-SQUARE-C Bus**

This feature allows FPGA configuration data to be downloaded from I-Square-C and stored into one of the three Configuration ROMs which are AT45DB642. U1, U3, and U24 have a separate Configuration ROM. This feature also allows rebooting (reloading) the FPGA. **CTPV2 FpgaConfigExamples.C shows the routines. When writing C code the first time, it is recommend to write and verify the C Code for the** Read FPGA Config Data from ROM to SRAM. This is to minimize the mistake of continuously Erasing and Programming the ROM. ROM is rated for 100,000 erases/writes cycles.

* 1. The overall steps are (details steps follow):
     1. Host issues Erase Commands to U1 to erase U1, U3, or U24.
     2. Host downloads the entire FPGA Configuration Data (MCS file) for U1, U3, or U24’s Configuration ROM to U1. U1 stores the data on onboard SRAM
     3. Host read back data from SRAM for verification.
     4. Host issues command to U1 to store data from SRAM into U1, U3, or U24’s Configuration. ROM. Host wait until U1 finishes storing data to ROM
     5. Host issues command to U1 to read Configuration ROM of U1, U3, or U24 to SRAM. Host wait for U1 to finish reading
     6. Host reads data from SRAM and verifies that data is corrected. If data does not match the MCS file, host can redo steps 8.1.1 to 8.1.6.
     7. If desire (not necessary) host can repeat steps 8.1.1 to 8.1.6 for the remaining FPGA.
     8. Host issues command to U3 to reboot ALL FPGA. There is no command to reboot individual FPGA.
  2. **Erase Command** (See Register File) Make sure U1 Register 17 (Status 2) bit 15 is a one:
     1. **For U1**
        1. Initialize variable Config\_ROM\_Block\_Number\_to\_Eraseto zero.
        2. Write Config\_ROM\_Block\_Number\_to\_Erase to U1 Register 7 bit 9..0
        3. Write the following bits values to U1 Register 6 to start Erase U1 Rom
           1. 10..9 = **01**
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **1001**
        4. Poll U1 Register 17 (Status 2) bit 15 until it is a one. Fpga Config is done erasing Config\_ROM\_Block\_Number\_to\_Erase.
        5. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = **01**
           2. 8 = **0**
           3. 7 = 0
           4. 6..3 = **1001**
        6. Increment Config\_ROM\_Block\_Number\_to\_Erase
        7. WAIT for 220 milliSecond. This is the time the ROM takes to Erase one block.
        8. Repeat 8.2.1.2 to 8.2.1.6 for Config\_ROM\_Block\_Number\_to\_Erase from 1 to 1023 to erase all 1024 ROM’s Blocks.
     2. **For U3**
        1. Initialize variable Config\_ROM\_Block\_Number\_to\_Eraseto zero.
        2. Write Config\_ROM\_Block\_Number\_to\_Erase to U1 Register 7 bit 9..0
        3. Write the following bits values to U1 Register 6 to start Erase U1 Rom
           1. 10..9 = **10**
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **1010**
        4. Poll U1 Register 17 (Status 2) bit 15 until it is a one. Fpga Config is done erasing Config\_ROM\_Block\_Number\_to\_Erase.
        5. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = **10**
           2. 8 = **0**
           3. 7 = 0
           4. 6..3 = **1010**
        6. Increment Config\_ROM\_Block\_Number\_to\_Erase.
        7. Repeat 8.2.2.2 to 8.2.2.6 for Config\_ROM\_Block\_Number\_to\_Erase from 1 to 1023 to erase all 1024 ROM’s Blocks.
     3. **For U24**
        1. Initialize variable Config\_ROM\_Block\_Number\_to\_Eraseto zero.
        2. Write Config\_ROM\_Block\_Number\_to\_Erase to U1 Register 7 bit 9..0
        3. Write the following bits values to U1 Register 6 to start Erase U1 Rom
           1. 10..9 = **11**
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **1011**
        4. Poll U1 Register 17 (Status 2) bit 15 until it is a one. Fpga Config is done erasing Config\_ROM\_Block\_Number\_to\_Erase.
        5. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = **11**
           2. 8 = **0**
           3. 7 = 0
           4. 6..3 = **1011**
        6. Increment Config\_ROM\_Block\_Number\_to\_Erase
        7. Repeat 8.2.3.2 to 8.2.3.6 for Config\_ROM\_Block\_Number\_to\_Erase from 1 to 1023 to erase all 1024 ROM’s Blocks.
  3. Download FPGA Config Data to SRAM (See Register File) Make sure U1 Register 17 (Status 2) bit 15 is a one:
     1. **For U1, U3, U24**
        1. Write the following bits values to U1 Register 6 to select SRAM for writing.
           1. 10..9 = 00
           2. 8 = 0
           3. 7 = **1**
           4. 6..3 = 0000
        2. Initialize 22 bits variable SRAM\_Address to 0
        3. Read 2 bytes from Xilinx MCS data bytes to variable SRAM\_DATA. The first byte read go to bits 15..8 and the second byte read go to bits 7..0.
        4. Write SRAM\_DATA to U1 Register 8
        5. Write bits 15..0 of SRAM\_Address to U1 Register 9.
        6. Write the following bits values to U1 Register 10 to write SRAM\_Data to SRAM\_Address.
           1. 15 = 1
           2. 14 = 0
           3. 5..0 = SRAM\_Address bits 21..16
        7. Write the following bits values to U1 Register 10 to ready for the next write.
           1. 15 = 0
           2. 14 = 0
           3. 5..0 = SRAM\_Address bits 21..16
        8. Increment SRAM Address
        9. Repeat 8.3.1.2 to 8.3.1.8 for remaining bytes in MCS file
  4. ReadBack FPGA Config Data to SRAM (See Register File) Make sure U1 Register 17 (Status 2) bit 15 is a one:
     1. **For U1, U3, U24**
        1. Write the following bits values to U1 Register 6 to select SRAM for reading.
           1. 10..9 = 00
           2. 8 = 0
           3. 7 = **1**
           4. 6..3 = 0000
        2. Initialize 22 bits variable SRAM\_Address to 0
        3. Read 2 bytes from Xilinx MCS data bytes to variable SRAM\_DATA. The first byte read go to bits 15..8 and the second byte read go to bits 7..0.
        4. Write bits 15..0 of SRAM\_Address to U1 Register 9.
        5. Write the following bits values to U1 Register 10 to read SRAM\_Data from SRAM\_Address.
           1. 15 = 0
           2. 14 = **1**
           3. 5..0 = SRAM\_Address bits 21..16
        6. Poll U1 Register 17 (Status 2) bit 15 until it is a one.
        7. Write the following bits values to U1 Register 10 to ready for the next write.
           1. 15 = 0
           2. 14 = 0
           3. 5..0 = SRAM\_Address bits 21..16
        8. Read Data From SRAM at U1 Register 11
        9. Compare Data From SRAM (8.4.1.8) to SRAM\_DATA (8.4.1.3)
        10. Increment SRAM Address
        11. Repeat 8.3.1.2 to 8.3.1.8 for remaining bytes in MCS file
  5. Program FPGA Config Data from SRAM to ROM (See Register File) Make sure U1 Register 17 (Status 2) bit 15 is a one:
     1. **For U1**
        1. Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM
           1. 10..9 = 01
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **0000**
        2. Poll U1 Register 17 (Status 2) bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM~~.~~ ~~Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding~~ the ROM (AT45DB642) maximum write of 100,000 times.
        3. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = 01
           2. 8 = 0
           3. 7 = 0
           4. 6..3 = **0000**
     2. **For U3**
        1. Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM
           1. 10..9 = **10**
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **0001**
        2. Poll U1 Register 17 (Status 2) bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM. ~~Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding~~ the ROM (AT45DB642) maximum write of 100,000 times.
        3. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = 10
           2. 8 = 0
           3. 7 = 0
           4. 6..3 = **0001**
     3. **For U24**
        1. Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM
           1. 10..9 = **11**
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **0010**
        2. Poll U1 Register 17 (Status 2) bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM. ~~Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding~~ the ROM (AT45DB642) maximum write of 100,000 times.
        3. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = 11
           2. 8 = 0
           3. 7 = 0
           4. 6..3 = **0011**
  6. Read FPGA Config Data from ROM to SRAM (See Register File) Make sure U1 Register 17 (Status 2) bit 15 is a one:
     1. **For U1**
        1. Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM
           1. 10..9 = 01
           2. 8 = 1
           3. 7 = 0
           4. 6..3 = **0011**
        2. Poll U1 Register 17 (Status 2) bit 15 until it is a one. One indicates that Fpga Config is done reading Config Data from ROM and storing to SRAM.
        3. Write the following bits values to U1 Register 6 to ready for next command
           1. 10..9 = 01
           2. 8 = 0
           3. 7 = 0
           4. 6..3 = **0011**
        4. Follow steps under ReadBack FPGA Config Data to SRAM to verify that data is the ROM match MCS file.
  7. Rebooting FPGA (Reloading FPGA with Config Data from ROM):
     1. Before issuing this command, make sure the Config Data for U1 is corrected because U1 code contains the Configuration algorithm. If wrong code is loaded into U1, a local reload (with a lab top) has to be done. This command will reboot all three FPGA
     2. **For all FPGA**
        1. Write the following bits values to U3 Register 7 to reboot all FPGA
           1. **2..0 = 101**
           2. **2..0 = 010**
           3. **2..0 = 101**

1. **Board Serial Number**
   1. The board serial number is stored in Serial ROM (SROM) in ASCII format. The board serial number is CTPV2xxx where xxx is (ASCII) number of the board. The serial ROM is connected to U3.
   2. Steps to read the board serial number
      1. Set SROM Address variable to zero.
      2. Write SROM Address variable to **U3** Config 2 bits 9..0
      3. Set Bit 15 of **U3** Config 2 to read one byte at SROM Address
      4. Reset Bit 15 of **U3** Config 2.
      5. Poll bit 15 of **U3** Status 3 for Read Done
      6. Read byte (in ASCII) from **U3** Status 3 bits 7..0.
      7. ReSet Bit 15 of **U3** Config 2.
      8. Increment SROM Address.
      9. Repeat 9.2.1 to 9.2.7 for the remain Bytes.
2. **Steps to Align Data from FLASH**
   1. **Set** control\_mgt bit 0. This Reset MGT Channels.
   2. **Reset** control\_mgt bit 0. Wait 1 uS for all channels to come up
   3. **ReSet** control\_mgt bit 1. This instruct FADC to send Counting Sequence to CTP
   4. **Set** control\_mgt register Bit 2 of all FLASH board. This allows FLASH board(s) to send alignment sequence to CTP.
   5. **Set** Config1 register Bit 3 of U1, U3, and U24 of CTPV2 board. This allows U1, U3, and U24 to execute alignment routine.
   6. Bring SYNC line to high to all boards for at least 1uS.
   7. Bring SYNC line back to low.
   8. **Reset** control\_mgt register Bit 2 of all FLASH board. This disables FLASH from sending alignment sequence at SYNC.
   9. **Reset** Config1 register Bit 3 of U1, U3, and U24. This disables U1, U3, and U24 from executing alignment routine at SYNC.
   10. Poll Status 1 Bit 9 of U24 for a one. This indicates the alignment is successful. If this bit does not go high after 200nS, there is something wrong.
   11. **Set** control\_mgt bit 1. This instruct FADC to send data from ADC FPGA to CTP
3. **Crate ID**
   1. The 16-bits Crate ID is to identify the location of the crate in a system. It is programmable by the user and sent to the SSP on the lower sixteen bits of the first 64-bits word
4. **Spectrometer Scalers Mode**
   1. The CTP scalers will be used for the online monitoring of the photon beam spectrum. The photon spectrum will be reconstructed by forming a coincidence between scintillators positioned in the electron and positron arms of the spectrometer. Each arm consists of 128 FADC250 channels. FADC250s of the electron/positron arm are placed in the left/right side of the VXS crate. To reconstruct the beam energy, we’ll need to read out the channel number for the electron and positron counters which are used to form a coincidence. The search of a coincidence is initiated by the rising edge of the FADC pulse. The FADC pulse (Hit Bit) from the FADC is high when the ADC sample is above a user configurable threshold. In the case when an electron or positron produce multiple hits in the spectrometer, only one hit per spectrometer arm is reported. The channel that is the first hit is reported. If there are simultaneous hits, the smallest channel is reported. Figure 2 illustrates the processing algorithm:
   2. The processing algorithm is as follow:
      1. When a hit (rising edge of HitBit) occurred in any of the left arm ADC channel, a Window (search/inhibit) is activated and that ADC channel number is stored in 8-bits register. While this Window is active, if a hit occurred in any of the right arm ADC channel, the 32-bits Coincident Counter is increment at the end of the left Window. Both channel numbers are captured in two separate registers for read out.
      2. When a hit (rising edge of HitBit) occurred in any of the right arm ADC channel, a Window (search/inhibit) is activated and that ADC channel number is stored 8-bits in register. While this Window is active, if a hit occurred in any of the left arm ADC channel, the 32-bits Coincident Counter is increment at the end of the right Window. Both channel numbers are captured in two separate registers for read out.
      3. There is only one 32-bits Coincident Counter.
      4. If more than one hit occurred at the same time, the lowest channel number is stored. The channel numbers are numbered from 1 to 128 for both left and right arm. The mapping of the channel number to crate slot number and ADC board channel is shown in Table 1.
      5. At the end of Window the 32-bits Hit Counter is increment respectively. The left arm has one 32-bit Hit Counter and the right arm has one 32-bit Hit Counter.
      6. The Window time is configurable from 4ns to 16 ns by configuring Config 3 bits 3 to 0.
      7. A 48-bit Timer Stamp increment at CLK rate (4nS).
      8. All the registers are clear when SYNC is high
      9. The time stamp and counters are latched for reading on rising edge of Config 3 bit 6. After the counters are latched, if Config 3 bit 7 is a one, all four counters will be cleared.
      10. The time stamp and counters registers can be read in any order.
      11. Steps to read and clear counter.
          1. Set Config 3 bits 7 (to clear counters) and 6 (to latch counters)
          2. Read Time Stamp and Counter registers
          3. Reset Config 3 bit 6 (to ready for next read).
      12. Steps to read and NOT clear counters
          1. Reset Config 3 bit 7 (Do not clear counters) and Set bit 6 (to latch counters)
          2. Read Time Stamp and Counters registers
          3. Reset Config 3 bit 6 (to ready for next read).
   3. Zeroes are sent to SSP
   4. To Select this mode, Reset Config 3 bit 15 to zero
5. **Spectrometer Scalers Counter Mode** 
   1. No processing in this mode
   2. 16 Hit Bits from one of the FLASH are sent to SSP bits 15..0
   3. The FLASH is fixed.
   4. To Select this mode, set Config 3 bit 15 to one.
6. Figure 3 and 4 show the firmware (VHDL code) block diagrams.
7. Table 1 and 2 show Hit Channel Number to Slot and FADC Channel Number Mapping for Left and Right Arm.
8. **Register File**

**I-Square-C Address Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **I-Square-C Board Address** | **I-Square-C Sub Address** | **R/W** | **Function** |
| **0 (U1)** | **0** | **R** | **Status 0**  **15: Payload 13 MGT Channel Up**  **14: Payload 11 MGT Channel Up**  **13: Payload 9 MGT Channel Up**  **12: Payload 7 MGT Channel Up**  **11: Undefine**  **10: Undefine**  **9: Payload 15 MGT Lane 1 Up**  **8: Payload 15 MGT Lane 0 Up**  **7: Payload 13 MGT Lane 1 Up**  **6: Payload 13 MGT Lane 0 Up**  **5: Payload 11 MGT Lane 1 Up**  **4: Payload 11 MGT Lane 0 Up**  **3: Payload 9 MGT Lane 1 Up**  **2: Payload 9 MGT Lane 0 Up**  **1: Payload 7 MGT Lane 1 Up**  **0: Payload 7 MGT Lane 0 Up** |
|  | **1** | **R** | **Status 1**  **15..2: Undefine**  **1: Payload 15,13,11,9,7 MGT Channels Up**  **0: Payload 15 MGT Channel Up** |
|  | **2** | **R/W** | **Config 0**  **6: Enable Payload 7**  **8: Enable Payload 9**  **10: Enable Payload 11**  **12: Enable Payload 13**  **14: Enable Payload 15** |
|  | **3** | **R/W** | **Config 1**  **3: Enable Alignment of Data from FLASH on falling edge of Sync**  **1: Init All MGT** |
|  | **4** | **R** | **Die Temperature** |
|  | **5** | **R** | **Vint** |
|  | **6** | **R/W** | **Config 2**  **Fpga Configuration:**  **10..9 = 1 -> Sel U1 for configuration**  **= 2 -> Sel U3 for configuration**  **= 3 -> Sel U24 for configuration**  **8 = Rising edge Execute Opcode**  **7 = 1 Select SRAM for writing**  **6..3 = 0 -> PROGRAM\_DATA\_U1.**  **1 -> PROGRAM\_DATA\_U3.**  **2 -> PROGRAM\_DATA\_U24.**  **3 -> READ\_DATA\_U1.**  **4 -> READ\_DATA\_U3.**  **5 -> READ\_DATA\_U24.**  **6 -> READ\_EPROM\_ID\_U1.**  **7 -> READ\_EPROM\_ID\_U3.**  **8 -> READ\_EPROM\_ID\_U24.**  **9 -> ERASE\_EPROM\_U1**  **A -> ERASE\_EPROM\_U3**  **B -> ERASE\_EPROM\_U24**  **C ->**  **D ->**  **E ->**  **F ->**  **2..0 = undefine** |
|  | **7** | **R/W** | **Config 3**  **Fpga Configuration:**  **9..0 -> Config ROM Block Number to Erase** |
|  | **8** | **R/W** | **Config 4**  **Sram Data**  **15..0 = Data to be written to SRAM** |
|  | **9** | **R/W** | **Config 5**  **Sram Address**  **15..0 = Sram Address 15..0** |
|  | **10** | **R/W** | **Config 6**  **Sram Address, R/W**  **15 = Rising edge write Sram Data to Sram Address**  **14 = Rising edge read Data at Sram Address. Data is available at Data From Sram**  **13..6 = undefined**  **5..0 = Sram Address 21..16** |
|  | **11** | **R** | **Status 3**  **Data From Sram**  **15..0 = Data read from Sram Address** |
|  | **12-16** |  | **Undefine** |
|  | **17** | **R** | **Status 2**  **15: 1 -> Fpga Config is Idle. Ready for opcode**  **14..0: Firmware Version** |
| **----------------** | **---------------** | **-------** | **--------------------------------------------------------** |
|  |  |  |  |
| **1 (U3)** | **0** | **R** | **Status 0**  **15: Payload 14 MGT Channel Up**  **14: Payload 12 MGT Channel Up**  **13: Payload 10 MGT Channel Up**  **12: Payload 8 MGT Channel Up**  **11: Undefine**  **10: Undefine**  **9: Payload 14 MGT Lane 1 Up**  **8: Payload 14 MGT Lane 0 Up**  **7: Payload 12 MGT Lane 1 Up**  **6: Payload 12 MGT Lane 0 Up**  **5: Payload 10 MGT Lane 1 Up**  **4: Payload 10 MGT Lane 0 Up**  **3: Payload 8 MGT Lane 1 Up**  **2: Payload 8 MGT Lane 0 Up**  **1: Payload 6 MGT Lane 1 Up**  **0: Payload 6 MGT Lane 0 Up** |
|  | **1** | **R** | **Status 1**  **15..2: Undefine**  **1: Payload 16,14,12,10,8 MGT Channels Up**  **0: Payload 16 MGT Channel Up** |
|  | **2** | **R/W** | **Config 0**  **7: Enable Payload 8**  **9: Enable Payload 10**  **11: Enable Payload 12**  **13: Enable Payload 14**  **15: Enable Payload 16** |
|  | **3** | **R/W** | **Config 1**  **3: Enable Alignment of Data from FLASH on falling edge of Sync**  **1: Init All MGT** |
|  | **4** | **R** | **Die Temperature** |
|  | **5** | **R** | **Vint** |
|  | **6** | **R/W** | **Config 2**  **SROM**  **15: Rising edge read byte from SROM Address**  **9..0: SROM Address** |
|  | **7** | **R/W** | **Config 3**  **2..0 = 5 -> reboot ALL FPGA** |
|  | **8-10** |  | **Undefine** |
|  | **11** | **R** | **Status 3**  **Data From Serial ROM**  **15: ASCII Data from SROM Valid**  **7..0: ASCII Data read from SROM Address** |
|  | **12=16** |  | **Undefine** |
|  | **17** | **R** | **Status 2**  **15: Undefine**  **14..0: Firmwarw Version** |
| **----------------** | **---------------** | **-------** | **--------------------------------------------------------** |
|  |  |  |  |
| **2 (U24)** | **0** | **R** | **Status 0**  **15: Payload 2 MGT Channel Up**  **14: Payload 5 MGT Channel Up**  **13: Payload 1 MGT Channel Up**  **12: Payload 3 MGT Channel Up**  **11: Payload 6 MGT Lane 1 Up**  **10: Payload 6 MGT Lane 0 Up**  **9: Payload 4 MGT Lane 1 Up**  **8: Payload 4 MGT Lane 0 Up**  **7: Payload 2 MGT Lane 1 Up**  **6: Payload 2 MGT Lane 0 Up**  **5: Payload 5 MGT Lane 1 Up**  **4: Payload 5 MGT Lane 0 Up**  **3: Payload 1 MGT Lane 1 Up**  **2: Payload 1 MGT Lane 0 Up**  **1: Payload 3 MGT Lane 1 Up**  **0: Payload 3 MGT Lane 0 Up** |
|  | **1** | **R** | **Status 1**  **15..9: Undefine**  **9: DataAlign Done**  **8: Error in Sum in Test Mode**  **7: Fiber Channel Ready**  **6: Fiber Lane Remote Up**  **5: Fiber Lane Byte Align**  **4: Fiber Lane Channel Align**  **3: History Data REady**  **2: Payload 6 MGT Channel Up**  **1: Payload 1,2,3,4,5,6 MGT Channels Up**  **0: Payload 4 MGT Channel Up** |
|  | **2** | **R/W** | **Config 0**  **0: Enable Payload 1**  **1: Enable Payload 2**  **2: Enable Payload 3**  **3: Enable Payload 4**  **4: Enable Payload 5**  **5: Enable Payload 6** |
|  | **3** | **R/W** | **Config 1**  **15..3:**  **3: Enable Alignment of Data from FLASH on falling edge of Sync**  **2: Reset Fiber MGT**  **1: Init All MGT**  **0: History Arm** |
|  | **4** | **R** | **U24 FX100T Die Temperature** |
|  | **5** | **R** | **U24 FX100T Vint** |
|  | **6** | **R/W** | **Undefine** |
|  | **7** | **R/W** | **Undefine** |
|  | **8** | **R/W** | **~~Final Sum Threshold LSB~~ History Buffer Trigger Enable Bit 15..0** |
|  | **9** | **R/W** | **~~Final Sum Threshold MSB~~ History Buffer Trigger Enable Bit 31..16** |
|  | **10** | **R** | **History Buffer Data LSB** |
|  | **11** | **R** | **History Buffer Data MSB** |
|  | **12** | **R/W** | **SD Test Control Register**  **Bits:**  **0 : Reset SYNC Count Reg 14**  **1 : Reset TRIG 1 Count Reg 15**  **2 : Reset TRIG 2 Count Reg 16** |
|  | **13** | **R** | **SD Clock Frequency (MHz)**  **Bits (7.. 0) : Clock 250 Count (Should be 250 +/- 2)** |
|  | **14** | **R** | **Count Rising Edge of Sync From SD**  **Bits (15:0) Count Rising Edge of SYNC from SD. Sync must be high for at least 30 nS** |
|  | **15** | **R** | **Count Rising Edge of Trig1 from SD**  **Bits (15:0) Count Rising Edge of TRIG1 from SD. Sync must be high for at least 30 nS** |
|  | **16** | **R** | **Count Rising Edge of Trig2 from SD**  **Bits (15:0) Count Rising Edge of TRIG2 from SD. Sync must be high for at least 30 nS** |
|  | **17** | **R** | **Status 2:**  **15: Undefine**  **14..0: Firmware Version** |
|  | **18** | **R/W** | **Config 2: Crate ID** |
|  | **19** | **R/W** | **Config 3:**  **15 🡪 1 Select Spectrometer Counter Mode**  **14..8 🡪**  **7 🡪 1: Clear all Counters after read status registers**  **6 🡪 rising edge latch time stamps and all counters.**  **3..0 🡪 Spectrometer Coincident Window Width / Hit Rate Count Inhibit Width** |
|  | **20** | **R/W** | **Config 4:**  **15..0 🡪** |
|  | **21** | **R/W** | **Config 5:**  **15..0 🡪** |
|  | **22** | **R/W** | **Config 6:**  **15..0 🡪** |
|  | **23** | **R/W** | **Config 7:**  **15..0 🡪** |
|  | **24** | **R/W** | **Config 8:**  **15..0 🡪** |
|  | **25** | **R/W** | **Config 9:**  **15..0 🡪** |
|  | **26** | **R** | **Status 3**  **15..00 🡪 TimeStamp Bit 15..0** |
|  | **27** | **R** | **Status 4**  **15..00 🡪 TimeStamp Bit 31..16** |
|  | **28** | **R** | **Status 5**  **15..00 🡪 TimeStamp Bit 47..32** |
|  | **29** | **R** | **Status 6**  **15..0 🡪 Left Arm Hit Counter Bit 15..0** |
|  | **30** | **R** | **Status 7**  **15..0 🡪 Left Arm Hit Counter Bit 31..16** |
|  | **31** | **R** | **Status 8**  **15..0 🡪 Right Arm Hit Counter Bit 15..0** |
|  | **32** | **R** | **Status 9**  **15..0 🡪 Right Arm Hit Counter Bit 31..16** |
|  | **33** | **R** | **Status 10**  **15..0 🡪 Coincident Hit Counter Bit 15..0** |
|  | **34** | **R** | **Status 11**  **15..0 🡪 Coincident Hit Counter Bit 31..16** |
|  | **35** | **R** | **Status 12**  **15..8 🡪 Right Hit Channel**  **7..0 🡪 Left Hit Channel** |
|  | **36** | **R** | **Status 13**  **15..0 🡪** |
|  |  |  |  |

The history data advance after Data MSB is read. Hence read LSB then MSB.

**Channel Number to PayLoad Mapping:**

|  |  |  |
| --- | --- | --- |
| **FPGA** | **FADC # Channel** | **Payload** |
| U24 | 0 | 3 |
|  | 1 | 1 |
|  | 2 | 5 |
|  | 3 | 2 |
|  | 4 | 4 |
|  | 5 | 6 |
|  |  |  |
| U1 | 0 | 7 |
|  | 1 | 9 |
|  | 2 | 11 |
|  | 3 | 13 |
|  | 4 | 15 |
|  |  |  |
| U3 | 0 | 8 |
|  | 1 | 10 |
|  | 2 | 12 |
|  | 3 | 14 |
|  | 4 | 16 |
|  |  |  |
|  |  |  |

Temperature\_C = ((float)TempRegValue \* 503.975/1024) - 273.15;

Front Panel LED:

+5V

D

C

B

A

6

5

4

3

C

B

A

2

1

0

LED 1 = U1

LED 0 = U3

LED2-6 = U24

PLL Lock

Trigger

Fiber OK

U24 Heart Beat

U3 Heart Beat

U1 Heart Beat

Data Alignment Done

Trig Monitor Lemo Connector is OR of 30 bits Output OF Hit Coincident WindowFront Panel LVDS IN, LVDS OUT:

-

+

Trig Out = Sum > Threshold

-

+

LVDS IN

LVDS OUT

1

8

P N

P N

Figure 2: Illustration of Processing of a hit bits pattern at CTP

Figure 3: Spectrometer Arm Processing

CH 36

0

3

2

1

FADC

ch=1..128

(VXS

left side)

FADC

ch=1..128

(vxs

right side)

Left Hit Ch : CH 1

Right Hit Ch : CH 25

CH 121

CH 25

CH 1

CH 120

CH 4

Left Window

Right Window

Left Hit Counter

0

3

2

1

Left Hit Counter

0

2

1

Coincident Counter

0

36

1

0

4

25

Clear Hit counter

1

5

r1

r3

r2

Reset

Window

Active

Window

r3

HitBits

Window

Active

r1

sync

Slot 1

Ch8

16

16

Q

Set

Reset

Window

Width

timer

Q

En

Ld

= 0

5

Priority

Encoder

Q

Priority

Encoder

Q

Latch

Priority

Encoder

Q

Window

Active

Channel

Hit

En

clear

counter

Q

sync

Hit

Counter

Figure 4: Spectrometer Processing

RD Clr Cnt En

Sync

RW3

RW3

RW2

RW1

W2

W1

RW1

Cl1

HC1

CH1

W1

Hit Counter

Window Active

Channel Hit

LEFT ARM PROCESSING

Reset Window Active

Clear Hit counter

Cl2

RW2

HC2

CH2

W2

Hit Counter

Window Active

Channel Hit

Right ARM PROCESSING

Reset Window Active

Clear Hit counter

Coincident Counter

Q

En

Clear

Rd Counter Req

Delay

Cl1

Cl2

LATCH

en

CH1

HC1

CH2

HC2

CH1\_Latch

HC1\_Latch

CH2\_Latch

HC2\_Latch

TS\_Latch

Table 1 : Left Arm Hit Channel Hit To Slot # Flash Channel # mapping

|  |  |  |
| --- | --- | --- |
| Left arm Hit Channel | Slot # | FADC Channel |
| 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 | 3 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 17,18,19,20,21,22,23,24,24,26,27,28,29,30,31,32 | 4 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48 | 5 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64 | 6 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80 | 7 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96 | 8 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112 | 9 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128. | 10 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |

Table 2 : Right Arm Hit Channel Hit To Slot # Flash Channel # mapping

|  |  |  |
| --- | --- | --- |
| Left arm Hit Channel | Slot # | FADC Channel |
| 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 | 13 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 17,18,19,20,21,22,23,24,24,26,27,28,29,30,31,32 | 14 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48 | 15 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64 | 16 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80 | 17 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96 | 18 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112 | 19 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |
| 113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128. | 20 | 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15 |

Appendix A:

Notes:

* Block Erase is used because Chip Erase Command is currently not supported by the IC AT45DB642
* Buffer To Memory Page Program without Built-in Erase is used because with Built-in Erase does not seems to work reliable.
* It takes 3199 pages of AT45DB642 memory to hold config. Data for FX70T. It takes 13 minutes presently to do this.